27th TJIMO
ALEXANDRIA, VIRGINIA

Round: **Power**

1 What is a Graph?

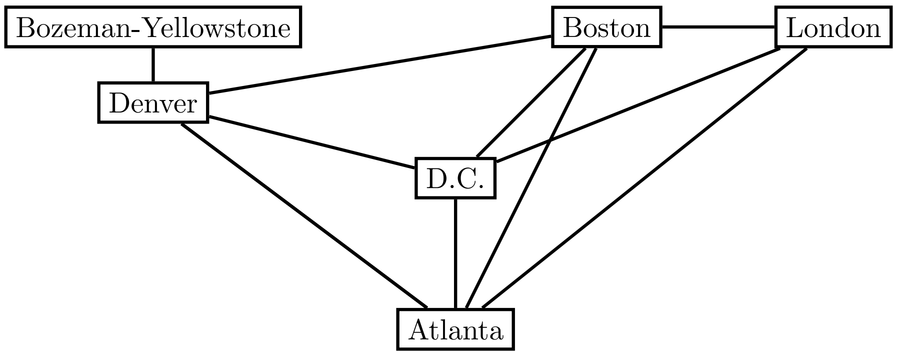
You might have heard of a graph in your math class as a plot with an x and y axis. Or maybe in science you looked at bar graphs and pie graphs. Maybe in history you took a look at line graphs. But the graphs we are going to talk about today are none of the above.

1.1 Some Introduction

At its core, a *graph* consists of two sets: a group of objects, and a group of connectors that each join two objects.

To gain insight into what a graph is, let's first consider flights that connect airports around the world. We depend on airplanes to travel quickly and efficiently. When we want to travel between major cities, like between D.C. and Boston, we can often take a single flight to make the trip. Enough people want to travel between these two cities for it to be economical to offer a single flight between them.

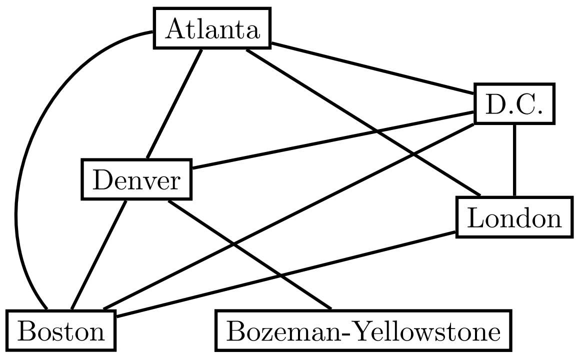
However, it is not always the case that single-flight trips are economically feasible, either due to lack of demand or due to needing to refuel. To travel from London to Yellowstone, for example, one must first stop in Atlanta, hop on a connecting flight to Denver, and then finally travel to Yellowstone. Denver and Atlanta are what we call hubs, since they are very large airports that accomodate heavy travel loads. To travel between smaller airports, we often need to stop at one of these larger hubs.



Our connections between our cities in the graph represent only direct connections between cities, but from our diagram, we can still see it is possible to travel between any two cities in our graph.

Neither the exact position of each city nor how connections might intersect matters. In other words, the way we draw a graph on paper does not change its properties, and for the most part are not useful to us. The following is an equivalent graph:

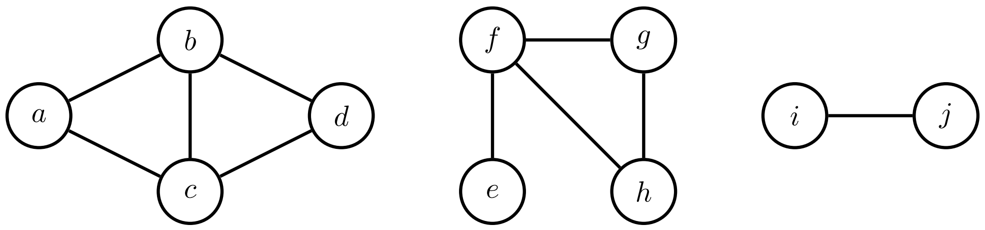
Time limit: 45 minutes.



The only characteristics of a graph we care about are the objects (in this case, cities) themselves, which we'll call *vertices*, and the connections joining the objects, which we'll call *edges*.

1.2 Definition of a Graph

We now have the background necessary to present the mathematical definition of a graph:
Definition 1. A *graph* $G(V, E)$ consists of a set of vertices V and a set of edges E.



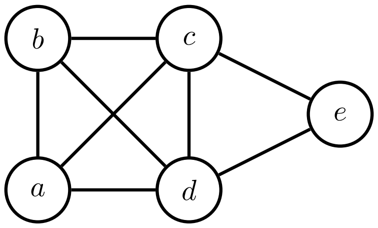
Definition 2. An *edge* is a collection of exactly two vertices. In the above graph, there is an edge between a and b, so $\{a, b\}$ is an edge. Note that $\{a, b\}$ is the same edge as $\{b, a\}$.
Definition 3. A *set* is a collection of any number of objects. In the above graph, we have the set of vertices

$$V = \{a, b, c, d, e, f, g, h, i, j\}.$$

We also have the set of edges

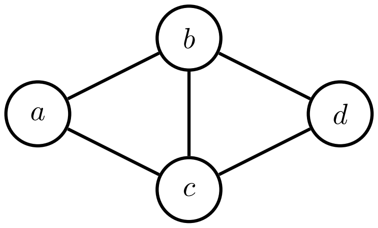
$$E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{b, d\}, \{a, c\}, \{e, f\}, \{f, g\}, \{g, h\}, \{h, f\}, \{i, j\}\}.$$

Definition 4. Given a set A, the cardinality $|A|$ denotes the number of elements in A.
Problem 1. Evaluate $|V|$ and $|E|$ for the sets V and E in Definition 3, the definition of the set. Note that $\{a, b\}$ counts as one single edge.
Solution. Answer: $|V| = |E| = \boxed{10}$.
There are 10 vertices and 10 edges in the graph.
Problem 2. Draw any graph with 5 vertices and 8 edges.
Solution. The following graph has 5 vertices and 8 edges.



1.3 Graph Basics

Definition 5. The *degree* of a vertex is the number of neighbors it has. For vertex b in the following graph, we write $deg(b) = 3$, meaning the degree of b is 3.



Problem 3. Show that the total number of edges in a graph is equal to

$$\frac{1}{2} \cdot \sum_{\text{all vertices } v} deg(v).$$

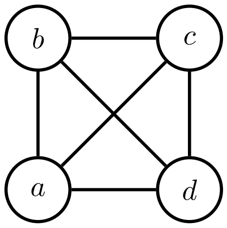
The symbol $\sum$ means add up $deg(v)$ over all vertices v. In the graph in Definition 5,

$$\begin{aligned}\sum_{\text{all vertices } v} deg(v) &= deg(a) + deg(b) + deg(c) + deg(d) \\ &= 2 + 3 + 3 + 2 \\ &= 10.\end{aligned}$$

Solution. The degree of a vertex represents the total number of edges that include that vertex. When we sum all the degrees together, we consider every edge in the graph twice, since each edge contributes to the degrees of two different vertices. Thus, the total number of edges in the graph is equal to

$$\frac{1}{2} \cdot \sum_{\text{all vertices } v} deg(v).$$

Definition 6. A *clique* is a graph such that there is exactly one edge covering every pair of distince vertices in the graph. We denote a clique of n vertices as K_n. Pictured is a clique of 4 vertices, or K_4.



Problem 4. Solve each of the following:

(a) How many edges are in the clique of 4 vertices, K_4?

(b) How many edges are in the clique of 5 vertices, K_5?

(c) In terms of n, how many edges are in the clique of n vertices, K_n, where n is any positive integer?

Solution. The number of edges in a clique are as follows:

(a) K_4 has $\boxed{6}$ edges: $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, d\}$, $\{b, d\}$, $\{c, d\}$.

(b) K_5 has $\boxed{10}$ edges: $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, d\}$, $\{b, d\}$, $\{c, d\}$, $\{a, e\}$, $\{b, e\}$, $\{c, e\}$, $\{d, e\}$.

(c) In general, K_n has

$$\binom{n}{2} = \boxed{\frac{n \cdot (n-1)}{2}}$$

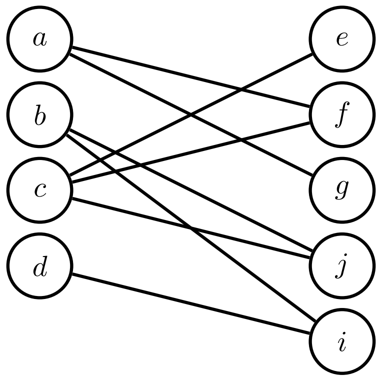
edges.

We need to count the total number of possible ways to choose two different vertices from n total vertices. We have n ways to choose the first vertex and $n - 1$ ways to choose the second vertex, but the order in which we choose the vertices doesn't matter (that is, $\{A, B\}$ represents the same edge as $\{B, A\}$), so we must divide by 2 to get the correct answer of $\frac{n \cdot (n-1)}{2}$.

Alternatively, we could have used the previous problem directly and plugged into our formula.

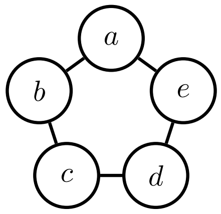
2 Bipartite Graphs

Definition 7. A *bipartite graph* is a graph whose vertices can be divided into two disjoint, or nonoverlapping, sets, S and T, such that every edge in the graph connects one vertex in S to one vertex in T. The following is an example of a bipartite graph:



We can split the vertices into two sets: $S = \{a, b, c, d\}$ and $T = \{e, f, g, h, i\}$. Every edge in the graph connects a vertex in S to one in T. No edge connects two vertices from S, and no edge connects two vertices from T.

Problem 5. Given a graph G, three vertices a, b, and c in the graph are such that an edge connects a and b, an edge connects b and c, and an edge connects c and a. Show that the graph G cannot be bipartite.
Solution. If G were bipartite, then its vertices can be partitioned into exactly two sets such that the vertices in each set are not connected by an edge. this means that a and b are in different sets, b and c are in different sets, and c and a are in different sets. however, this is not possible with only two sets, so g must not be bipartite.
Problem 6. As a result of the previous problem, bipartite graphs have no K_3. However, given a graph G with no K_3, must it be bipartite?
Solution. No. The following graph has no K_3, but is not bipartite.



If it were bipartite, a would be in a different set as b, so it would be in the same set as c, and so on, meaning a is in a different set as a, a contradiction.
Problem 7. Let n be an even integer. What is the maximum number of edges in a bipartite graph with n vertices? Express your answer in terms of n, and remember to explain your solution! You might find the following fact useful:
Theorem 1 (AM-GM)**.** *Given two nonnegative numbers x and y,*

$$\frac{x+y}{2} \geq \sqrt{x \cdot y}.$$

Solution. Since the graph is bipartite, we can separate the graph into two sets, S and T, such that $|S| + |T| = n$, and no edges connect each set's vertices. The maximum number of edges in a graph separated into S and T is $|S| \cdot |T|$. However, by the AM-GM Inequality,

$$|S| \cdot |T| \leq \left(\frac{|S| + |T|}{2}\right)^2 \leq \boxed{\frac{n^2}{4}}.$$

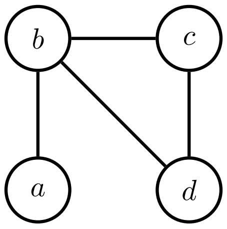
We see that this number of edges can be achieved by separating the n vertices into two equally-sized sets.

3 Shortest Paths

Graphs represent connections between different objects. We might, for example, have different cities connected in some way. It takes a certain amount of time to get in between different neighboring cities. Perhaps to travel from D.C. to St. Louis, we must either stop at Chicago or Cincinnati along the way. It might be faster to travel through Cincinnati, but it might be cheaper to stop at Chicago. Either way, we need to devise some kind of system to determine for us the best way to travel between vertices in our graph, through following some kind of *path.*

3.1 Introduction to Paths

Definition 8. A *path* is a sequence of vertices, such that every two consecutive vertices in the graph are connected by an edge. All edges in the path must be distinct, and all the vertices must be distinct.
Problem 8. In the following graph, answer the questions below.



(a) Is b, c a path?

(b) Is b, c, d a path?

(c) Is a, c, b a path?

(d) Is b, c, b a path?

(e) Is b, c, d, b a path?

(f) Is b a path?

If your answer for any of these was "no," explain why.
Solution. From Definition 8 of a path,

(a) b, c is a path.

(b) b, c, d is a path.

(c) a, c, b is not a path, since a and c are not connected.

(d) b, c, b is not a path, since b is repeated.

(e) b, c, d, b is not a path, since b is repeated.

(f) b is a path.

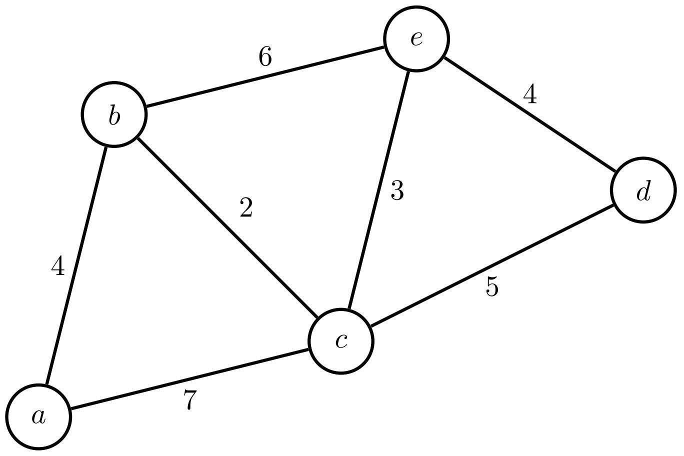
3.2 Dijkstra's Shortest Path Algorithm

In this subsection we explore an algorithm to compute the shortest, or cheapest, path from one vertex to another in a graph. Thus, we'll want to assign some kind of value to each edge. For example, it might be the distance between two vertices. We'll call this value the *weight*.
Definition 9. We assign to each edge $\{u, v\}$ a nonnegative *weight* $w(u, v) \geq 0$. This is simply a number associated with the edge. For example, it might represent the distance between u and v or the cost to travel between u and v.

Our main goal is to find a path from one vertex to another such that the sum of the weights along the path is minimized. To travel along an edge in the path, we have to pay the weight of that edge.

Definition 10. Given a graph G, the *shortest path* v_0, v_1, v_2, v_3, $\cdots$, v_{m-2}, v_{m-1}, v_m from vertex v_0 to vertex v_m is the path that minimizes the sum

$$w(v_0, v_1) + w(v_1, v_2) + w(v_2, v_3) + \cdots + w(v_{m-2}, v_{m-1}) + w(v_{m-1}, v_m).$$



In the above graph, the shortest path from a to e is a, b, c, e. The shortest, or cheapest, distance from a to e is then 9. No other path can do better.

Problem 9. Determine the shortest path and shortest-path length of each vertex from a. List the vertices in increasing order of the distance from a to that vertex.

Solution. Shortest paths, in order:
a: a; length 0
b: a, b; length 4
c: a, b, c; length 6
e: a, b, c, e; length 9
d: a, b, c, d; length 11

Definition 11. A *source* is a vertex from which paths begin. In the above problem, a was the source. Dijkstra's algorithm is a single-source algorithm, which means it computes the shortest path from one source to all other vertices.

Problem 10. Let u be a vertex immediately prior to vertex v along the shortest path from source s to v. Show that the shortest path from s to v must pass along a shortest path to u before traversing one final edge from u to v.

Solution. Suppose for the sake of contradiction that the path to u along the shortest path to v is not the shortest path. This means there exists a path to u shorter than the one along the path to v. But then there must also exist a shorter path to v, along the shortest path to u first and then along the edge from u to v, which is a contradiction. Then the path to u along the path to v is a shortest path.

Dijkstra's algorithm involves keeping track of a best known distance, which we'll call $dist(v)$ for each vertex v. Initially, $dist(v) = \infty$ for all vertices v, except for the source s of our graph, for which $dist(s) = 0$.

Then, if we were to run the algorithm on the example graph from before, the *dist* of each vertex is as follows, at the very beginning:

vertex v	$dist(v)$
a	0
b	∞
c	∞
d	∞
e	∞

As we explore our graph, $dist(v)$ decreases, until $dist(v)$ represents the actual shortest-path distance from a source to v.

The way we explore the graph is we visit vertices one by one, in the order of closeness to the source, as you have computed in Problem 9. Therefore, the first vertex we visit is the source itself.

We only visit a vertex v if we know for sure that the actual shortest-path distance to v matches what we have stored in $dist(v)$. When we visit v, we take a look at all vertices u which share an edge with v. If $dist(v)+w(v,u) < dist(u)$, then we change the value of $dist(u)$ to be $dist(v)+w(v,u)$. In other words, we have explored enough of our graph to discover a path through v to u that is shorter than the one known up to that point. This means that $dist(v)$, which is the best known distance thus far, also represents the best path to v through all visited vertices.

Notice how, in the table below, $dist(b)$ and $dist(c)$ are updated, as they are neighbors of a, which was just visited.

vertex v	$dist(v)$	visited?
a	0	yes
b	4	no
c	7	no
d	∞	no
e	∞	no

The vertex that has the minimum *dist* that we haven't visited yet is b. Notice how $dist(c)$ and $dist(e)$ are updated in the table below, as they are neighbors of b.

vertex v	$dist(v)$	visited?
a	0	yes
b	4	yes
c	6	no
d	∞	no
e	10	no

$dist(b) = 4$, which agrees with the actual shortest path distance to b.

Problem 11. What is the next vertex we should visit? (Hint: which unvisited vertex has the minimum *dist*?) Copy the table in your answer and update it so that three vertices are visited. Does *dist* of this vertex agree with the actual shortest path distance to it, as you computed in Problem 9?

Time limit: 45 minutes.

Solution. The next vertex we visit should be c.

vertex v	$dist(v)$	visited?
a	0	yes
b	4	yes
c	6	yes
d	9	no
e	11	no

$dist(c) = 6$, which agrees with the actual shortest path distance to c.

What you should notice is that $dist(v)$ is the actual shortest distance to v for all visited vertices v.

Problem 12. Suppose that we have just visited k vertices, where $k > 0$ is some positive integer. Furthermore, suppose that these k vertices are the k closest of all the vertices to the source. Show that, of all the unvisited vertices, the vertex v with the minimum $dist(v)$ is such that $dist(v)$ is equal to the actual shortest distance from the source to v. In other words, show that we can visit this vertex and make it the $(k+1)$-th visited vertex.

Solution. $dist(v)$ represents the best distance from the source to v using only visited vertices. Suppose $dist(v)$ wasn't actually the least distance from the source to v. Then the path must have gone through some unvisited vertex. Call that vertex u. However, we assumed v to be the unvisited vertex with the least $dist$, so $dist(u) \geq dist(v)$. However, this means the length of the path through u is at least as long as $dist(v)$. Therefore, $dist(v)$ must be equal to the least distance from the source to v.

The previous problem allows us to continue to visit the closest unvisited vertex, until we have visited all the vertices. Once we finish visiting all the vertices, $dist(v)$ will represent the shortest distance from the source to v. Dijkstra's algorithm in its entirety is outlined below!

Algorithm 1 Dijkstra

```
for all vertices $v$ do
    $dist(v) \leftarrow \infty$                          ▷ This means set $dist(v)$ to be $\infty$.
    $visited(v) \leftarrow 0$                            ▷ This means set $visited(v)$ to be 0.
    $prev(v) \leftarrow -1$              ▷ This allows us to trace our path backwards to the source.
$dist(src) \leftarrow 0$                          ▷ Initialize the value of $dist$ for the source.
while there exists a vertex $v$ such that $visited(v) = 0$ do
    Let $v$ be the vertex such that $visited(v) = 0$ that minimizes $dist(v)$
    $visited(v) \leftarrow 1$                                          ▷ Visit the vertex.
    for all neighbors $u$ of $v$ do
        if $visited(u) = 0$ then
            $alt \leftarrow dist(v) + weight(v, u)$
            if $alt < dist(u)$ then
                $dist(u) \leftarrow alt$      ▷ If we found a different path, update the value of $dist$.
                $prev(u) \leftarrow v$
```

Dijkstra's algorithm lets us comput the shortest or cheapest path from one vertex to another extremely quickly – roughly on the order of the number of edges. This means that even in a graph with 100,000 vertices and 100,000 edges a computer program can compute the shortest distance from one vertex to another in a fraction of a second! Perhaps in high school, you will code this up yourself and see it in action.

4 Eulerian Circuits

Definition 12. A *walk* is a sequence of vertices in which every pair of consecutive vertices is connected by an edge. Note that this is different from a path: walks are allowed to repeat vertices.
Definition 13. An *Eulerian circuit*, named after the Swiss mathematician Leonhard Euler, is a walk that begins and ends on the same vertex and traverses every edge in the graph exactly once.
Problem 13.

4.1 Introduction to Cycles

4.2 Seven Bridges of Königsberg